VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Mobilní interaktivní průvodce hradem Sovinec

Interactive Guide to Castle Sovinec

2020 Dominik Namyslo

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dominik Namyslo**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Mobilní interaktivní průvodce hradem Sovinec
Interactive Guide to Castle Sovinec

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem bakalářské práce je vyvinout aplikaci sloužící jako interaktivní průvodce po hradě Sovinec. Řešení by mělo obsahovat klasický textový průvodce doplněný o multimediální obsah, navigaci po památce a zobrazování obsahu v kontextu aktuálního umístění (lokalizace pomocí GPS, případně míst zájmů označených QR kódem). Aplikace by pro zatraktivnění volitelně mohla využít i prvky rozšířené reality (AR). Cílovou platformou budou mobilní zařízení s operačním systémem Android.

Řešení bude obsahovat:
1. Rešerši českých a zahraničních aplikací sloužících jako průvodci po památkách na platformě Android s důrazem na využité technologie a poskytované funkce.
2. Návrh formátu a způsobu uložení multimediálních dat s ohledem na snadnou editaci, případně budoucí doplnění obsahu.
3. Implementaci vlastní aplikace (s podporou i18n).
4. Otestování aplikace na dostupných zařízeních a emulátorech s různými verzemi OS Android.
5. Závěrečné shrnutí dosažených výsledků a srovnání s konkurenčními produkty.

Multimediální obsah (texty, obrázky, zvuky) bude dodán. Práce je řešena ve spolupráci s kastelánem hradu Sovinec.

Seznam doporučené odborné literatury:

[1] Reto Meier, Ian Lake, Professional Android, Wrox; 4th edition, 2018, ISBN 978-1118949528
[2] Neil Smyth, Android Studio 3.0 Development Essentials - Android 8 Edition, CreateSpace Independent Publishing Platform, 2017, ISBN 978-1977540096
[3] John Horton, Android Programming for Beginners, Packt Publishing, 2015, ISBN 978-1785883262

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 15. května 2020

Namyslka Dominik

Rád bych poděkoval vedoucímu bakalářské práce Mgr. Ing. Michalovi Krumniklovi, Ph.D. za jeho odbornou pomoc při návrhu a implementaci této práce a dále zaměstnancům Muzea v Bruntále a dobrovolníkům na hradě Sovinec za jejich zpětnou vazbu při testování výsledné aplikace.

Abstrakt

Cílem bakalářské práce je vytvoření mobilní aplikace pro platformu Android sloužící jako virtuální průvodce hradem Sovinec. Součástí práce je i webová aplikace potřebná pro administraci obsahu a serverová část zajišťující komunikaci mezi mobilní aplikací a administrátorským rozhraním. Aplikace bude vhodně zobrazovat obsah v textové formě doplněný o obrázky a audio-průvodce. Tento obsah bude aplikace nabízet v několika různých jazycích. Aplikace bude rovněž klást důraz na snadnou manipulaci s jejím obsahem, bez nutnosti zásahu do zdrojového kódu.

Výsledná aplikace nemá za cíl nahradit klasického průvodce hradem. Cílem aplikace je zejména poskytnout doprovodný obsah uživatelům, kteří nechtějí využít služeb průvodce, ale zároveň se chtějí dozvědět zajímavé informace o veřejně přístupných částech hradu.

Klíčová slova: Android, Node.js, TypeScript, React, GraphQL, aplikace, průvodce, hrad Sovinec

Abstract

The purpose of this bachelor thesis is to create a mobile application for the Android platform, that will serve as a virtual guide through castle Sovinec. A web application necessary for content administration and server-side application ensuring communication between the mobile application and administration interface are also part of this thesis. The application will suitably display text and images as well as provide an audio guide. The content of the mobile application will be accessible in multiple different languages. The application also needs to emphasize easy manipulation with the content without the need for change in application source code.

Finished application should not serve as a replacement for in-person castle guide. The main aim of the application is to provide additional content to the users that do not wish to use guide services but still want to learn interesting information about publicly accessible parts of the castle.

Keywords: Android, Node.js, TypeScript, React, GraphQL, application, guide, Sovinec Castle

Obsah

Seznam použitých zkratek a symbolů **9**

Seznam obrázků **10**

Seznam výpisů zdrojového kódu **11**

1 Úvod **12**

2 Rešerše konkurenčních aplikací **13**
2.1 Srovnávané aplikace 13
2.2 Souhrn 17

3 Návrh **19**
3.1 Fyzické limitace 19
3.2 Forma obsahu 19
3.3 Funkce aplikace 19

4 Architektura **22**
4.1 Komunikace 22

5 Použité technologie **24**
5.1 React 24
5.2 Node.js 24
5.3 TypeScript 25
5.4 GraphQL 25

6 Implementace **26**
6.1 Serverová část 26
6.2 Webová část 27
6.3 Mobilní část 30
6.4 Oprávnění 32

7 Testování **36**

8 Nasazení **37**

9 Závěr **38**

Literatura **39**

Přílohy **40**

A Příklad souboru Docker Compose **41**

B Licence **42**

Seznam použitých zkratek a symbolů

GPS	–	Global Positioning System
NFC	–	Near field communication
API	–	Application Programming Interface
REST	–	Representational State Transfer
SQL	–	Structured Query Language
CRUD	–	Create, Read, Update, Delete
IDE	–	Integrated Development Environment
DOM	–	Document Object Model
I/O	–	Input/output
CPU	–	Central Processing Unit
SDL	–	Schema Definition Language
UI	–	User Interface
CSS	–	Cascading Style Sheets
XML	–	Extensible Markup Language
QR	–	Quick Response
ID	–	Identification
APK	–	Android Package
DevOps	–	Development and Operations
SDK	–	Software development kit
iOS	–	iPhone Operating System

Seznam obrázků

1 Snímky z aplikací Průvodce Mikulovem a Lewes Castle 15
2 Snímky z aplikace Orava Castle 16
3 Snímky z aplikace Bled Castle 17
4 Návrhy vzhledu mobilní aplikace 20
5 Architektura systému 22
6 Obrázková galerie 34

Seznam výpisů zdrojového kódu

1 Příklad stylování React komponent s použitím knihovny Emotion 29
2 Příklad vytvoření nové notifikace 30
3 Zjednodušený příklad souboru docker-compose.yaml 41

1 Úvod

V době, ve které vládnou světu mobilní zařízení, je pro provozovatele kulturních institucí stále složitější odpoutat pozornost návštěvníků od jejich telefonů a směrovat ji za účelem vzdělání a kulturního obohacení. To platí zejména pro návštěvníky mladších věkových skupin. Jednou z možností, jak zájem návštěvníků nasměrovat požadovaným směrem, je poskytnout jim obsah právě přes jejich mobilní zařízení. Pro tuto motivaci vzniká i tato aplikace. Cílem je využít možnosti chytrých telefonů a vyvolat v návštěvnících zájem absolvovat prohlídku s průvodcem. V následujících kapitolách je popsán kompletní postup při tvorbě této aplikace.

Ještě před zahájením samotné implementace je potřeba nejprve důkladně prozkoumat aplikace podobného charakteru. To je důležité zejména pro získání uceleného přehledu o tom, jaké funkce tyto aplikace obvykle nabízí. Získané poznatky můžeme použít jako inspiraci při návrhu vlastní aplikace a zároveň nám mohou pomoci vyvarovat se často opakovaným chybám. To vše slouží jako dobrý základ pro vývoj aplikace, která je uživatelsky přívětivá a obsahuje dostatečné množství funkcí pro probuzení zájmu v budoucích uživatelích.

Dalším krokem je provést analýzu možných řešení s ohledem na požadované funkce. Pro snadnější představu toho, jak bude aplikace vypadat a fungovat, je vhodné si v této části připravit několik náhledů budoucí aplikace. Současně je potřeba zaměřit pozornost na to, jakým způsobem bude funkcionalita systému zajištěna po architektonické stránce, a jakým způsobem budou probíhat persistence dat potřebných pro chod systému.

Teprve po provedení nezbytné analýzy může začít práce na samotné implementaci. Ta by měla začít vystavením komunikačního rozhraní na straně serveru. Následně můžeme přistoupit k implementaci klientů, kteří budou toto komunikační rozhraní využívat. Postup sice není nutno dodržet přesně v tomto pořadí, ale vyhneme se tím nutnosti vytvářet náhradu za skutečné rozhraní a nebudeme tak stavět systém založený na nereálných datech.

Výslednou aplikaci je vhodné před spuštěním pro veřejnost nejprve důkladně otestovat na co možná největším počtu zařízení. Tím je možné odhalit chyby, které se při samotné implementaci nemusí vůbec projevit. To je dáno především obrovským množstvím Android zařízení, které jsou na trhu dostupné. Pro zajištění optimální protestovanosti napříč platformou Android je vhodné, aby testovaná zařízení zastupovala všechny verze Androidu podporované finální aplikací.

2 Rešerše konkurenčních aplikací

Ještě před začátkem analýzy samotného projektu je vhodné si nejprve důkladně vyzkoušet aplikace s podobným zaměřením. Cílem je získat co možná nejpřesnější obrázek o tom, jak takové aplikace vypadají, jak se ovládají a jaké funkce obvykle svým uživatelům nabízí.

Při srovnání těchto aplikací je vhodné se zaměřit na jejich kladné i záporné stránky. To může pomoci s včasným odhalením chyb, jež snižují uživatelský dojem z těchto aplikací. Dle získaných poznatků můžeme následně upravit návrh vytvářeného systému a tím zajistit, aby se výsledný produkt stejným chybám vyvaroval.

2.1 Srovnávané aplikace

Pro účely srovnání došlo k nainstalování a následně k důkladnému odzkoušení přibližně patnácti různých aplikací pro platformu Android. Některé z těchto aplikací se svým zaměřením mohou mírně odchylovat od zaměření vytvářené aplikace. Takové aplikace mohou sloužit například jako průvodci celými městy nebo jako vyhledávače zajímavých výletních destinací.

Navzdory odlišnému účelu těchto aplikací si z nich i přes to můžeme odnést mnoho důležitých poznatků, jež nám následně mohou pomoci v návrhu cílové aplikace. Souhrn kladných a záporných stránek těch nejzajímavějších z nich je popsán níže.

2.1.1 Hrady a zámky (Česká republika) [1]

Jednou ze zajímavých funkcí, kterou aplikace nabízí, je obrázková galerie. Velkou výhodou galerie v této aplikaci je především funkce umožňující přiblížení obrázku. To je velmi žádoucí funkcionalita, zejména vezmeme-li v úvahu, že cílovou skupinou této aplikace jsou mobilní zařízení, která jsou obvykle vybavena displeji poměrně malých velikostí.

Obsah aplikace je tvořen zejména textem, jež je doplněn o obrázkovou galerii. Mimo to aplikace nabízí ještě možnost zahrát si kvíz, což může sloužit jako dobrý nástroj pro zlepšení dojmu z aplikace, jelikož samotný obsah aplikace nemusí některé uživatele dostatečně zaujmout. Před absolvováním kvízu je možné si zvolit jeho téma a obtížnost. Následuje několik otázek, po jejichž zodpovězení je kvíz vyhodnocen. Kvíz má však jeden poměrně velký nedostatek. Aplikace totiž ukazuje celkové hodnocení kvízu, ale již neukazuje, které z otázek byly zodpovězeny správně a které špatně. Uživatel tedy nemá možnost se ze svých chyb poučit.

Co bych aplikaci dále vytkl, je poměrně nezvyklé rozložení vysouvatelné navigace. Ta se na rozdíl od většiny ostatních aplikací vysouvá z pravé strany. Taková navigace může nejen vést k nechtěnému zmatení uživatele, ale rovněž porušuje doporučení Material Design [2], které slouží jako příručka pro jednotný vývoj aplikací na platformě Android [1].

[1] Aplikace Hrady a zámky je dostupná z https://play.google.com/store/apps/details?id=cz.simopt.amos.hrady

[2] Doporučení Material Design je dostupné z https://material.io

2.1.2 Průvodce Mikulovem (Česká republika) [3]

Jednou ze zajímavých funkcí, kterou aplikace poskytuje, je zobrazení mapy. Mapa umožňuje zobrazit zajímavá místa v okolí Mikulova a následně se o nich dozvědět detailnější informace. Mimo to je na mapě rovněž možno zobrazit některou z předem naplánovaných tras a využít mapu k navigaci mezi jednotlivými body, což lze vidět na obrázku 1a.

Stejně, jako u aplikace Hrady a zámky, je i zde možnost zobrazení obrázkové galerie. Ačkoliv galerii samotnou považuji za přínosnou funkcionalitu, způsob její implementace má však v této aplikaci jednu zásadní nevýhodu. Obrázky v galerii totiž není možné přiblížit. Velikost zobrazeného obrázku je tak napevno určena velikostí displeje zařízení. To ovšem na mobilních telefonech s malým displejem neumožňuje zobrazit dostatečné množství detailů.

Další zajímavou funkcí je možnost zobrazit kalendář s kulturními akcemi z okolí Mikulova. Při výběru některé z akcí se na mobilním zařízení otevře webový prohlížeč s detailními informacemi. Zde je škoda, že tato funkce není integrována přímo do samotné aplikace. Dalším možným vylepšením by bylo provázat aplikaci s kalendářem mobilního zařízení. Vybrané akce by si tak uživatel mohl například rovnou přidat do svého kalendáře.

2.1.3 Lewes Castle (Velká Británie) [4]

Aplikace slouží jako náhrada za klasického audioprůvodce při prohlídce hradu a přilehlého muzea. Obsah je rozdělen na několik zájmových bodů, které jsou doplněny o zvukové nahrávky. Ty je možno přehrát vybráním příslušného zájmového bodu z nabídky aplikace nebo využitím automatického přehrávání. Detail zájmového bodu s přehrávačem je zobrazen na obrázku 1b.

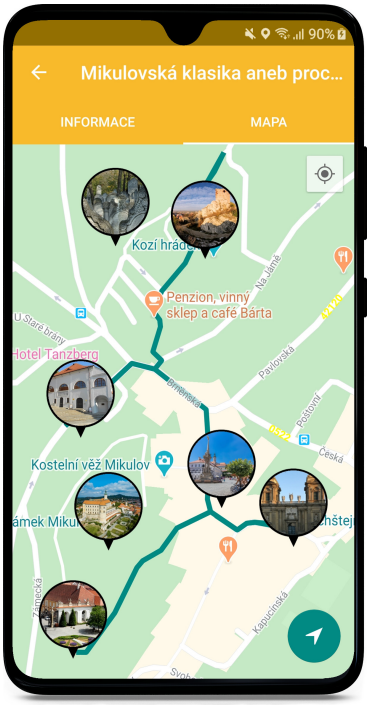
Pro zajištění funkcionality automatického přehrávaní jsou využity informace o poloze mobilního zařízení. Mobilní zařízení s aplikací je možno při prohlídce ponechat v kapse a přehrávání zvukových souborů je řízeno na základě jeho polohy. To však vyžaduje co nejpřesnější určení polohy uživatele. Přesnost geolokace je totiž do značné míry ovlivněna vnějšími vlivy a výsledná odchylka od skutečné polohy může být i v řádech desítek metrů.

Samotné zpracování zvukových nahrávek považuji za nadmíru povedené, jelikož využívají binaurální technologii a navíc jsou doplněny o doprovodné zvukové efekty. To může výrazně přispět ke vtažení posluchače do děje, a tím mu zprostředkovat lepší zážitek. Bohužel mimo možnost přehrát audioprůvodce toho aplikace již mnoho nenabízí.

Zde spatřuji několik možností, jak aplikaci vylepšit. První z nich je přidání doprovodného textu k zájmovým bodům. V současnost totiž jednotlivé body poskytují pouze možnost přehrát zvukovou nahrávku. To znamená, že uživatel bez sluchátek, popřípadě uživatel se sluchovým postižením není schopen aplikaci jinak využít. Aplikaci by rovněž velmi prospělo přidání dalšího multimediálního obsahu, jako jsou například obrázky.

[3] Aplikace Průvodce Mikulovem je dostupná z https://play.google.com/store/apps/details?id=cz.reinto.tcmikulov

[4] Aplikace Lewes Castle je dostupná z https://play.google.com/store/apps/details?id=asterixlab.echoesxyz.quynhnd.lewescastle



(a) Zobrazení mapy v aplikaci Průvodce Mikulovem

(b) Zobrazení přehrávače v aplikaci LewesCastle

Obrázek 1: Snímky z aplikací Průvodce Mikulovem a Lewes Castle

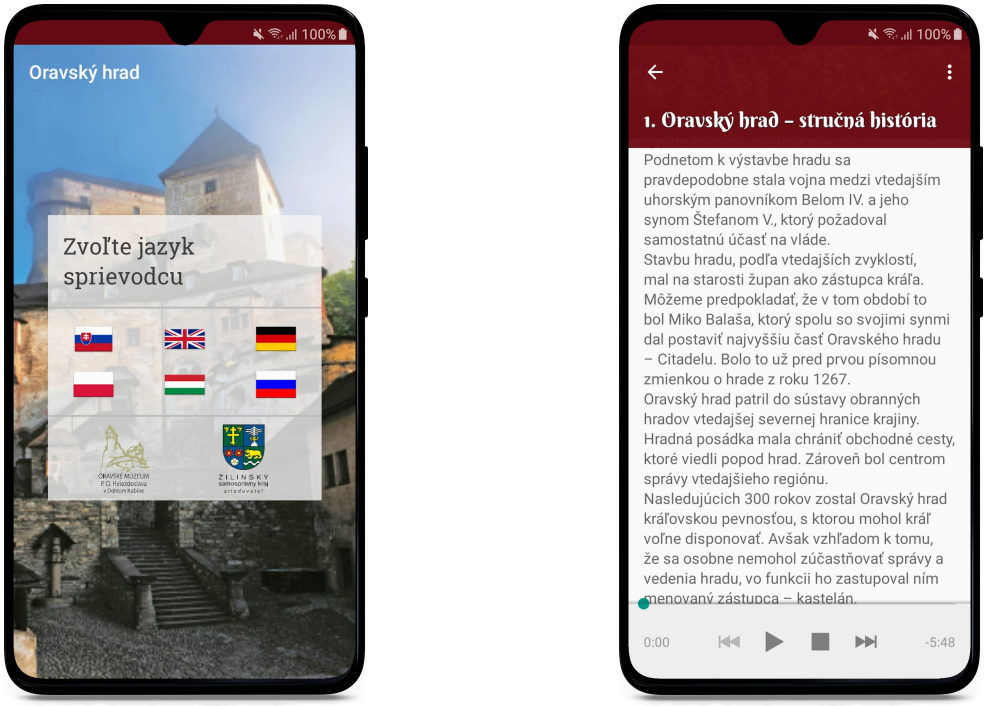
2.1.4 Orava Castle (Slovenská republika) [5]

Jednou z velkých výhod, které aplikace nabízí, je možnost zvolit si jeden z šesti dostupných jazyků. Přínos aplikace tak mohou naplno využít i cizojazyční návštěvníci hradu. Volba jazyku je zachycena na obrázku 2a. S volbou jazyku je spojen i způsob ukládání dat potřebných pro provoz aplikace. Zde vývojáři dle mého názoru zvolili jedno z nejlepších možných řešení, a sice to, že aplikace stahuje potřebná data až na základě zvoleného jazyka. Nedochází tak k hromadění nepotřebných dat a zbytečnému zabírání úložiště mobilního zařízení.

Jednotlivé zájmové body jsou doplněny o titulní obrázek, po jehož rozkliknutí se zobrazí obrázková galerie. Bohužel i v této aplikaci chybí u galerie jakákoliv možnost obrázek přiblížit. Zde bych navíc vytkl, že nikde není zobrazen počet obrázků, jež galerie obsahuje. Na první pohled se tedy může uživatelům jevit, že galerie obsahuje pouze jediný obrázek.

Co se týče samotného obsahu jednotlivých zájmových bodů, zde vidím poměrně značný prostor pro vylepšení. Aplikace nabízí možnost přehrát audioprůvodce, což vnímám jako velké plus. Na druhou stranu zobrazený text je místy poměrně dlouhý a celkově působí velice stroze. To může některé uživatele od čtení odradit. Tento dojem je navíc umocněn poměrně nevýrazným členěním textu na jednotlivé odstavce, což je dobře vidět na obrázku 2b.

[5] Aplikace Orava Castle je dostupná z https://play.google.com/store/apps/details?id=com.goodrequest.audioguide.orava



(a) Volba jazyka

(b) Detail zájmového bodu

Obrázek 2: Snímky z aplikace Orava Castle

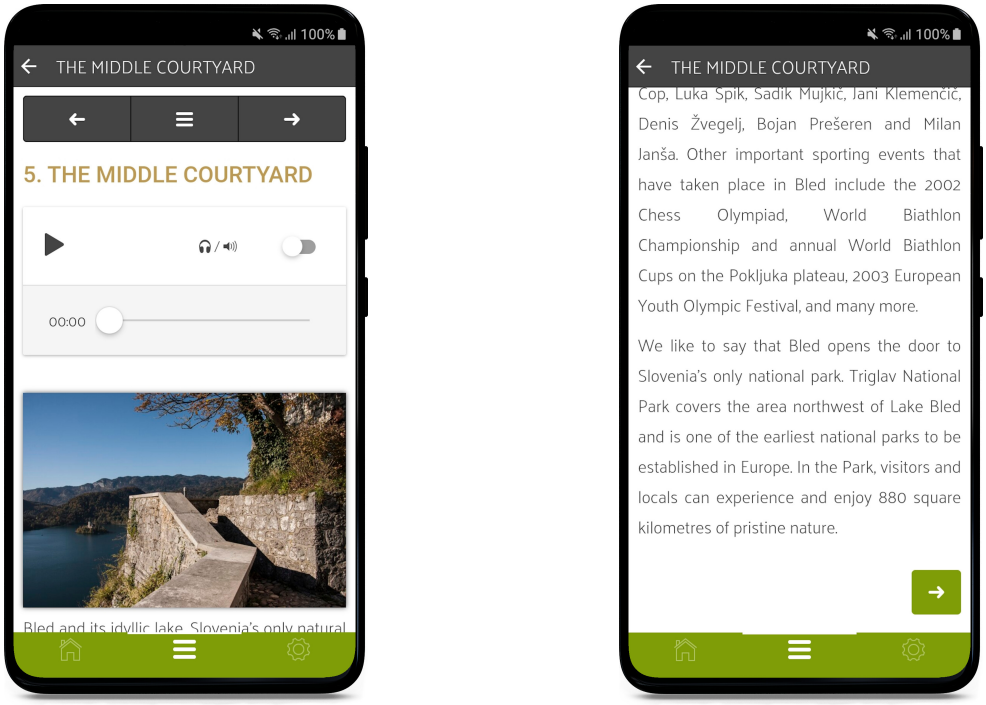
2.1.5 Bled Castle Guide (Slovinská republika) [6]

I tato aplikace umožňuje volbu z několika jazyků. Při testování této funkcionality jsem však objevil několik zásadních nedostatků. V některých případech se po zvolení jiného jazyka z nabídky změna vůbec neprojevila. Když už se povedlo jazyk změnit, nastal další problém. U některých jazyků byly přeloženy pouze texty a zvukové nahrávky nikoliv, u jiných tomu bylo zase naopak. Celkově chválím snahu vývojářů nabídnout uživatelům volbu z více jazyků, ale způsob, jakým je tato funkcionalita naimplementována, byl pro mě velmi frustrující.

Dalším z nedostatků této aplikace je špatné uspořádání navigace. Ta se zobrazuje hned na několika místech současně. Jedna z navigací se zobrazuje v horní části obrazovky, další v dolní části obrazovky a při zobrazení detailu zájmového bodu jsou navíc zobrazeny ještě navigační položky na začátku a na konci obsahu. Toho si lze povšimnout na obrázcích 3a a 3b, kde jsou zachyceny celkem čtyři různé navigační prvky. Takové uspořádání působí velice matoucím dojmem. Spousta funkcionality v zobrazených navigacích je navíc duplikovaná. Tento problém je možné vyřešit zvolením jednotného navigačního principu a jeho následným konzistentním dodržováním napříč celou aplikací.

[6] Aplikace Bled Castle Guide je dostupná z https://play.google.com/store/apps/details?id=si.virtualni_vodic.BledCastle

Na závěr bych u této aplikace rád zmínil způsob, jakým ukládá potřebná data. Audio soubory i obrázky jsou uloženy do veřejného externího úložiště. To znamená, že všechny tyto soubory jsou viditelné pro ostatní aplikace i pro mediální skener, a tudíž jsou zobrazeny i v galerii a hudebním přehrávači mobilního zařízení. Tento způsob mi osobně dost vadil, jelikož galerie byla zahlcena velkým množstvím nepotřebných obrázků.



(a) Detail zájmového bodu - horní část

(b) Detail zájmového bodu - dolní část

Obrázek 3: Snímky z aplikace Bled Castle

2.2 Souhrn

Jedním z často se opakujících prvků napříč srovnávanými aplikacemi je možnost přehrávání zvukových nahrávek. U některých aplikací byly nahrávky doplněny o hudební podkres nebo nejrůznější zvukové efekty. V takových nahrávkách vidím poměrně velký potenciál, jelikož kvalitně zpracovaná zvuková nahrávka dokáže uživatele snadno vtáhnout do děje.

Zde je potřeba zmínit, že samotné zvukové nahrávky by neměly být jedinou formou obsahu, kterou aplikace nabízí. Audioprůvodce má sloužit zejména jako doprovodný obsah určený ke zpestření zážitku z prohlídky hradu. Obsah aplikace by měl vždy být doplněn o textovou část a ideálně i o obrázky. Jedině tak je schopna uživatelům zprostředkovat dostatečný zážitek i za předpokladu, že si nemohou zvukové nahrávky poslechnout.

Ačkoliv jistá forma obrázkové galerie byla přítomna téměř ve všech srovnávaných aplikacích, častým nedostatkem byla absence funkcí sloužících pro přiblížení obrázků. To považuji z hlediska

uživatelského komfortu za jednu z klíčových funkcionalit, jelikož bez ní není možné na malém displeji mobilního zařízení zobrazit dostatečné množství detailů.

Jedna z testovaných aplikací využívá data o poloze zařízení k automatickému přepínání audioprůvodce. Tento přístup je však potřeba důkladně zvážit. Zájmové body by od sebe musely být v dostatečně velké vzdálenosti na to, aby šlo přesně určit, u jakého bodu se zrovna uživatel nachází. V opačném případě by mohlo docházet k nechtěnému přepínání mezi zájmovými body. Osobně jsem toho názoru, že aplikace by sama od sebe neměla zájmové body přepínat. To by mohlo vést k dezorientaci uživatele. Domnívám se, že vhodnější variantou pro využití informací o poloze je uživateli pouze nabídnout zobrazení obsahu na základě jeho polohy. Uživatel si tak bude moct sám zvolit, zda si přeje obsah zobrazit nebo ne.

Při testování konkurenčních aplikací mne překvapilo, že téměř žádná z těchto aplikací se nesnaží uživatele interaktivně zapojit. Jedním ze způsobů, jak uživatelům zprostředkovat interaktivnější dojem z aplikace, může být například přidání jednoduché minihry. V jedné ze srovnávaných aplikací byl tento prvek implementován formou kvízu. Kvíz může být vhodným řešením zejména u aplikací vzdělávacího charakteru, jelikož dokáže uživatele zabavit a zároveň je přimět k procvičení získaných vědomostí. Zde je však podstatné, aby kvíz poskytoval uživatelům zpětnou vazbu. Vždy by tak mělo dojít k zobrazení správných odpovědí na jednotlivé otázky. To je důležité proto, aby se uživatel následně mohl ze svých chyb poučit.

Poměrně zajímavou funkcí, kterou lze uživatelé nalákat k návštěvě hradu, je poskytnout jim přehled konaných akcí. Ty se mohou konat přímo na hradě nebo v blízkém okolí. Při implementaci této funkcionality může být výhodné využít funkcí kalendáře mobilního zařízení. Díky tomu by si mohl uživatel do svého osobního kalendáře jednoduše přidat akce, které chce navštívit.

3 Návrh

Jedním z kroků při tvorbě jakékoliv aplikace by měla vždy být její důkladná analýza. K tomu nám jako dobrý základ poslouží již provedená rešerše konkurenčních aplikací. Cílem analýzy je získat obrázek o tom, jak by měla výsledná aplikace vypadat, s jakými daty bude pracovat a jaké funkce bude svým uživatelům nabízet.

3.1 Fyzické limitace

Při návrhu funkcionality mobilní aplikaci došlo i k osobní návštěvě hradu Sovinec. Cílem této návštěvy bylo zjistit informace o fyzické dispozici hradu a probrat s jeho představiteli počáteční představy o tom, jak bude aplikace využívána. To rovněž pomohlo odhalit několik specifikací, se kterými je nutné počítat při návrhu funkcí aplikace a jejich následné implementaci.

Orientačním měřením přesnosti GPS v okolí hradu bylo zjištěno, že odchylka naměřené polohy mobilního zařízení se pohybuje maximálně v jednotkách několika metrů. Zde je samozřejmě potřeba počítat i s jistou rezervou, jelikož přesnost určované polohy je dána především možností výhledu na oblohu a může být negativně ovlivněna vnějšími vlivy.

Při této návštěvě vyšlo najevo, že v okolí hradu není k dispozici mobilní signál, a tedy ani přístup k mobilním datům. Jediné místo na hradě, kde je možné získat připojení k internetu, je na prvním hradním nádvoří. Zde je návštěvníkům umožněno připojení k veřejné Wi-Fi síti.

3.2 Forma obsahu

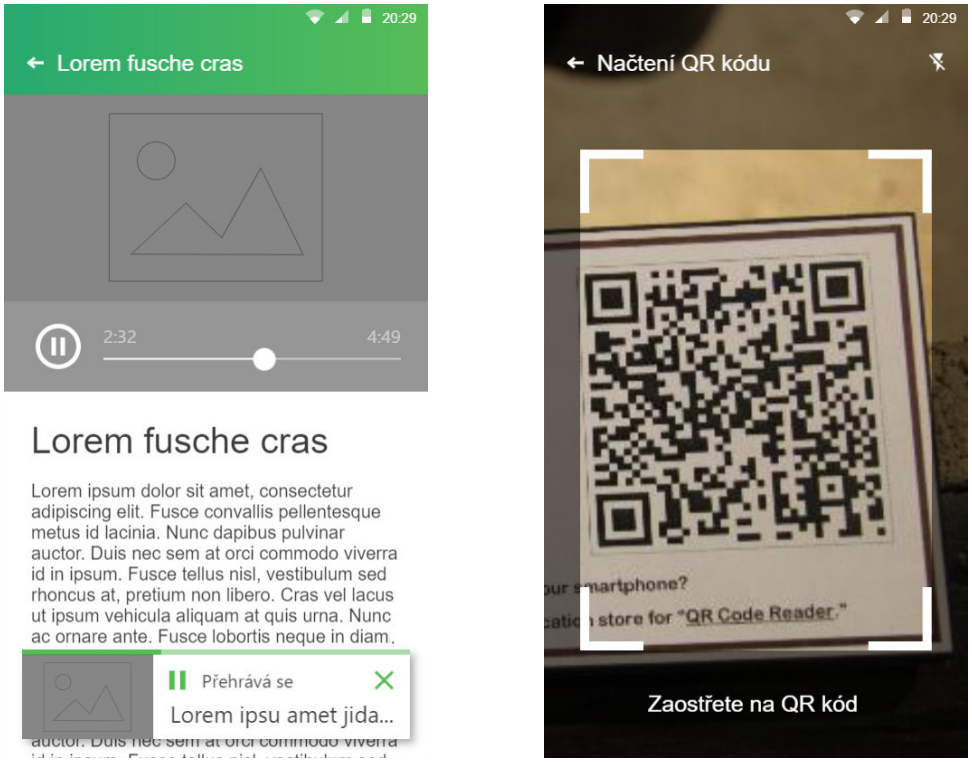
Bezesporu nejdůležitějším prvkem každé aplikace je právě její obsah. Většina konkurenčních aplikací nabízí obsah formou prostého textu, který je následně doplněn například o obrázkovou galerii. Důvod pro využití takové formy obsahu je zřejmý, z hlediska implementace se totiž jedná o jedno z nejjednodušších možných řešení. Jeho nevýhodou je však to, že samotný text může některým uživatelů připadat poměrně nezajímavý či dokonce nudný.

Proto jsem se u tvořené aplikace rozhodl využít jiný způsob tvorby textu. Obsah aplikace bude tvořen upravenou formou Rich textu. Ten se od prostého textu odlišuje v tom, že poskytuje členění na odstavce, může se volně prolínat s obrázky, umožňuje vkládat do textu odkazy a nabízí možnost základního formátování textu, jako např. ztučnění či podtržení.

Další formou obsahu, kterou bude aplikace nabízet, jsou zvukové nahrávky sloužící jako audioprůvodce. Tyto nahrávky budou doplňovat běžný obsah, nemusí však být dostupné u všech bodů hradu.

3.3 Funkce aplikace

Součástí každého návrhu je prvotní představa o funkcích, jež by měla aplikace nabízet. Velké množství těchto funkcí je samozřejmě ovlivněno rozborem konkurenčních aplikací. Jiné funkce jsou navrženy na základě fyzické dispozice hradu nebo přímo vyplývají z obsahu aplikace.



(a) Návrh ovládacích prvků (b) Návrh čtečky QR kódů

Obrázek 4: Návrhy vzhledu mobilní aplikace

3.3.1 Funkce navržené na základě obsahu aplikace

Jak již bylo zmíněno, obsah aplikace bude tvořen upravenou formou Rich textu. Tento text je potřeba následně nativně zobrazit v samotné aplikaci. Této funkce si zřejmě většina uživatelů ani nevšimne, nicméně její implementace je kritická pro poskytování obsahu aplikace.

Další formou obsahu, kterou by měla aplikace nabízet, je možnost přehrání audio průvodce. Z toho vyplývá několik funkcí, jež bude muset aplikace zajišťovat. Mimo samotné přehrávání audio souborů je také potřeba, aby aplikace poskytovala základní ovládací prvky. Uživatel tak bude moci přehrávání např. pozastavit a následně opět spustit. Návrh vzhledu ovládacích prvků pro přehrávání audioprůvodce je možné vidět na obrázku 4a.

3.3.2 Funkce navržené na základě fyzické dispozice

Jednou z funkcí navržených na základě fyzické dispozice hradu je způsob, jakým bude muset aplikace získávat data potřebná pro svůj provoz. Aby bylo možné zajistit, že aplikace bude schopna sloužit svému účelu i bez připojení k internetu, je nezbytné, aby se veškerá data potřebná k jejímu chodu dopředu stáhla a uložila v zařízení.

Aplikace může rovněž využít přesného určování polohy a na jejím základě nabídnou uživateli relevantnější obsah. Zde je třeba opatrnost a nepostavit všechny funkce aplikace na základě polohy mobilního zařízení. Někteří uživatelé totiž nemusí aplikaci udělit přístup ke své poloze.

3.3.3 Funkce navržené na základě konkurenčních aplikací

Aplikace bude nabízet obsah doplněný o obrázky. V konkurenčních aplikacích byly často přítomny i obrázkové galerie. To by měla nabízet i tato aplikace. Zde je potřeba klást důraz na to, aby galerie poskytovala funkci umožňující přiblížení obrázku.

V některých konkurenčních aplikacích je možné, aby si uživatel zvolil jazyk obsahu. To je vhodný prostředek k tomu, jak do aplikace zapojit i cizojazyčné návštěvníky. Je však potřeba klást důraz na to, aby nedošlo k přeložení pouze některých částí aplikace.

Další zajímavou funkcí, která by mohla být zapracována po vzoru konkurenčních aplikací, je vědomostní kvíz. Ten umožní návštěvníkům ověřit si své znalosti získané při prohlídce hradu. Kvíz zároveň pomůže s navázáním interakce mezi uživatelem a aplikací.

3.3.4 Dodatečné funkce

Pro zajištění orientace na hradě bude aplikace využívat princip načítání QR kódů. Ty by měly být rozmístěny v prostorech areálu hradu, a po jejich načtení by mělo dojít k zobrazení detailních informací vázaných k danému místu nebo např. ke konkrétnímu exponátu. Aplikace tedy bude muset zajistit funkci čtečky QR kódů. Návrh jejího vzhledu je vyobrazen na obrázku 4b.

Další funkcí ke zlepšení orientace na hradě je možnost využití technologie NFC. Ve štítku s QR kódem by mohl být rovněž zabudovaný NFC tag. Uživatel by si tak mohl vybrat, zda načte QR kód pomocí čtečky, nebo k němu pouze přiloží svůj mobilní telefon. Ačkoliv NFC tagy jsou dnes již poměrně levnou záležitostí, je zde potřeba počítat s jistou investicí.

4 Architektura

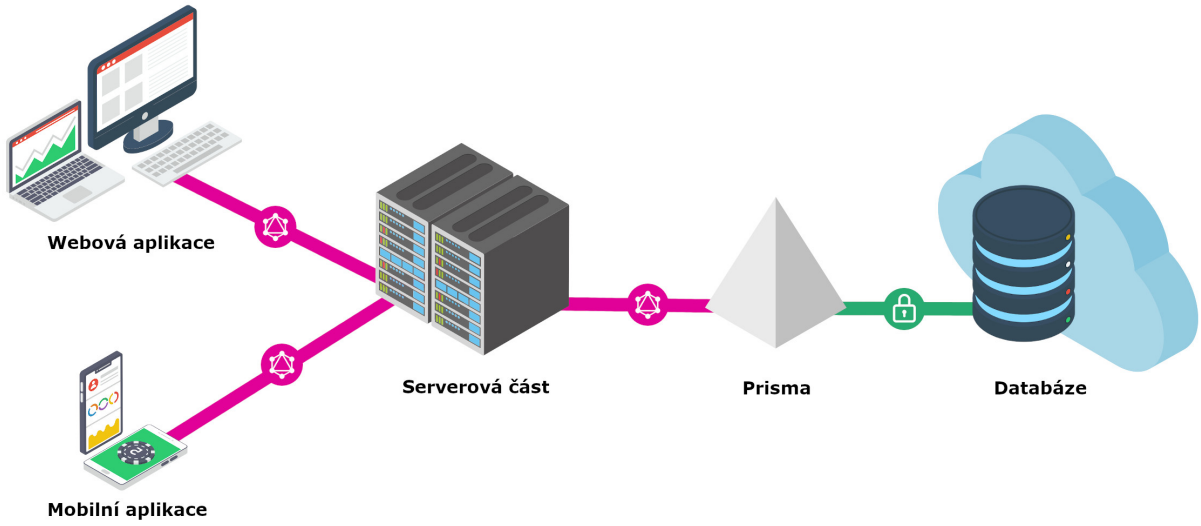
Jednou z požadovaných vlastností výsledného systému je také to, aby veškerý obsah aplikace bylo možno jednoduše spravovat, bez nutnosti zásahu do jejího zdrojového kódu. Pro zajištění takové funkcionality je mimo samotnou mobilní aplikaci navíc potřeba ještě obstarat perzistenci využívaných dat a administrátorské rozhraní sloužící k jejich manipulaci.

Pro trvalou perzistenci dat je v tomto případě nejvhodnější použít některý z relačních databázových systémů. Zde je samozřejmě třeba myslet i na to, že bude potřeba zprostředkovat komunikaci mezi databází, mobilní aplikací a webovým administrátorským rozhraním. Součástí systému tedy bude muset být i serverová část, která bude tuto komunikaci zajišťovat.

Zde je potřeba také zmínit, že systém bude muset zajistit perzistenci pro jednoduché datové typy, jako např. řetězce či čísla, i pro různá multimediální data. Pro ukládání multimediálních dat není relační databáze zcela vhodným řešením. Vhodnějším řešením je ukládání těchto dat přímo na disk a následně do databáze uložit pouze cestu k danému souboru. Perzistence multimediálního obsahu bude tedy zajištěna serverovou částí systému.

4.1 Komunikace

Všechny části systému mezi sebou musí určitým způsobem komunikovat. Tuto komunikaci můžeme rozdělit na dva větší celky. Prvním je komunikace mezi serverovou částí a jejími klienty, tedy webovou aplikací a mobilní aplikací. Druhým celkem je pak komunikace mezi serverovou částí a databázovým systémem. Kompletní architektura výsledného systému, včetně způsobu komunikace mezi jeho jednotlivými částmi, je znázorněna na obrázku 5.



Obrázek 5: Architektura systému

4.1.1 Komunikace mezi serverovou částí a klienty

Pro zajištění komunikace mezi serverovou částí a klienty se obvykle vystavuje komunikační API na serverové části systému. Při tvorbě API se v dnešní době nejčastěji využívá architektury postavené na principech REST. Zásadní nevýhodou takové architektury je značně rostoucí složitost implementace s každou další expanzí systému.

Proto jsem se rozhodl využít technologii GraphQL, která mimo jiné zajišťuje snadnější škálovatelnost výsledného systému. Hlavní výhody této technologie jsou popsány v kapitole 5.4. Serverová část tedy bude vystavovat GraphQL koncový bod, přes který bude probíhat veškerá komunikace mezi serverovou částí a jejími klienty.

4.1.2 Komunikace mezi serverovou částí a databází

Na závěr je potřeba ještě zajistit komunikaci mezi serverovou částí systému a databází. Komunikaci je samozřejmě možné zajistit manuálně s využitím jazyku SQL. Toto řešení je však z mého pohledu zbytečně komplikované. Je totiž potřeba zajistit velké množství repetitivních úkonů, jako např. parsování dat nebo ošetření případných chybových stavů. To vede u projektů využívajících větší množství databázových operací k výraznému natažení výsledného kódu.

Z těchto důvodů jsem se zde rozhodl využít open source framework Prisma. Ten dokáže na základě schématu databáze vygenerovat všechny CRUD operace nad daným schématem. Všechny operace jsou samozřejmě typované, takže s použitím správného IDE se můžeme snadno vyvarovat chybám, které by se jinak projevily až při běhu aplikace [2].

5 Použité technologie

Komplexní systémy často využívají velké množství nejrůznějších technologií. Některé tyto technologie jsou staré již mnoho let a za tu dobu jsou prověřeny nespočtem vyvinutých aplikací. To však neznamená, že představují nejefektivnější způsob práce, dokonce v některých případech může docházet k jejich využívání z obyčejné setrvačnosti. Starší technologie jsou značně limitovány svou architekturou, a tedy často nemohou nabídnout vývoj odrážející nejnovější trendy.

Cílem této práce je vytvoření a nasazení funkčního systému, který bude do jisté míry rovněž sloužit jako technologické demo předvádějící využití moderních technologií při vývoji reálného systému. V následujících kapitolách jsou stručně popsány nejzajímavější z těchto technologií.

5.1 React

React je knihovna pro JavaScript určená k budování komponent pro uživatelská rozhraní. Samotná technologie je oproštěna od způsobu, jakým jsou jednotlivé komponenty vykreslovány. Toho může dosáhnout tak, že si interně udržuje tzv. virtuální DOM. Ten je následně vykreslovací metodou přetransformován na reálný DOM nebo na jiný výstup, jenž je specifický pro konkrétní platformu. Z toho vyplývá, že záměnou vykreslovací metody může být React snadno adaptován pro použití na libovolné platformě [3].

Hlavním principem této technologie je atomický přístup k vývoji uživatelského rozhraní. Ten se opírá o vývoj atomických prvků, tzv. komponent, ze kterých se následně skládají komplexní prvky uživatelského rozhraní. Atomické komponenty jsme schopni vyvíjet v odděleném prostředí, které je nezávislé na zbytku systému. Výsledné komponenty jsme schopni rovněž samostatně otestovat a následně využít napříč celým systémem. To vede vývojáře k důslednějšímu dodržovaní principů pro opětovnou použitelnost kódu [4].

5.2 Node.js

Node.js je multiplatformní JavaScript prostředí s otevřeným zdrojovým kódem, jehož jádro je stejné jako u webového prohlížeče Google Chrome. Node.js aplikace běží na jediném procesu bez toho, aniž by docházelo k vytváření nového vlákna pro každý požadavek. Namísto toho je poskytnuta sada asynchronních I/O operací. Při provádění I/O operací tedy nedochází k blokování vláken a zbytečnému plýtvání CPU zdrojů. Node.js naváže na provádění požadavku poté, co dojde k provedení I/O operace a navrácení jejího výsledku [5].

Typickou překážkou při tvorbě webových aplikací je poměrně úzká specializace samotných vývojářů. Jednou z největších výhod Node.js je eliminace různých programovacích jazyků pro vývoj klientské a serverové části webových aplikací. Díky tomu odpadají tradiční role frontend a backend vývojářů, kdy vývoj obou částí je schopný plně zastoupit jediný vývojář se znalostí JavaScriptu [5].

5.3 TypeScript

Jednou z velmi populárních technologií spojených se současným vývojem v JavaScriptu je tzv. TypeScript. Ten není určen jako náhrada za klasický JavaScript, ale slouží pouze jako jeho nadstavba. Cílem technologie TypeScript je poskytnout chybějící nebo špatně podporovanou funkcionalitu a zároveň nabídnout nástroje k zajištění důslednější typové kontroly. Z toho také vyplývá, že jakýkoliv validní JavaScript kód je rovněž validní TypeScript kód [6].

Před tím, něž budeme schopni spustit TypeScript kód v JavaScriptovém prostředí, je zapotřebí nejprve provést jeho kompilaci. To může působit jako velká nevýhoda oproti klasickému JavaScriptu, kde k žádné kompilaci nedochází. Opak je však pravdou. Zdrojový kód je nejprve zkompilován do dobře podporované verze čistého JavaScriptu. Díky tomu může TypeScript nabídnout i funkcionalitu, kterou jinak nelze plně využít, například kvůli špatné podpoře napříč různými JavaScriptovými prostředími.

5.4 GraphQL

Technologie GraphQL představuje moderní přístup k implementaci API. Základem této technologie je dotazovací jazyk a běhové prostředí určené k vykonání jednotlivých dotazů. Samotné GraphQL představuje pouze komunikační standard, který není nijak vázán na konkrétní způsob implementace [7].

API, postavené na technologii GraphQL, poskytuje oproti starším formám API několik zásadních výhod. První z nich je, že GraphQL požadavky mohou přesně specifikovat, které z polí chtějí vrátit. Nedochází tak ke zbytečnému přenosu dat, které nejsou nijak využity. Další velkou výhodou je schopnost zanořování požadavků. To nám umožňuje získat data, u kterých je pro jejich získání potřeba využít vazeb mezi objekty s pomocí jediného dotazu a vyhnout se tak např. složitému spojování několika tabulek. GraphQL API poskytuje větší flexibilitu s ohledem na její možné budoucí rozšíření. Rozšíření API o nová pole nijak neovlivní stávající implementaci klientů využívajících toto API [8].

6 Implementace

Po vyhotovení rozboru konkurenčních aplikací, přehledu požadovaných funkcích a celkového obrazu o architektuře vyvíjeného systému přejdeme k samotné implementaci. Jelikož každá část systému se zabývá poměrně specifickými úkony, rozdělil jsem popis implementace na tři hlavní celky, a to na serverovou část, webovou část a mobilní část.

6.1 Serverová část

Hlavním úkolem serverové části je zprostředkování komunikace mezi webovou částí, mobilní částí a databází, která zajišťuje trvalé uchování dat. Serverová část je postavena na technologii Node.js, což nám umožní vyvíjet serverovou aplikaci s využitím jazyku JavaScript. Klasický JavaScript jsem navíc doplnil o technologii TypeScript, která přináší striktnější typovou kontrolu, a jejíž hlavní výhody jsem popsal v kapitole 5.3. Striktnější typová kontrola je vhodná pro budování robustnějších aplikací. Zde nám rovněž ve spojení s technologií GraphQL pomůže zamezit potřebě parsovat příchozí data. Můžeme se tak spolehnou na to, že máme vždy k dispozici všechna požadovaná data. Velkou výhodou je i podpora jazyku TypeScript ve vývojových prostředích. Kvalitní IDE je schopné nás již v průběhu psaní kódu upozornit na případné chyby, které mohou vzniknou nesrovnalostí v datových typech.

6.1.1 Prisma

Pro zajištění komunikace mezi serverovou částí a databází využívá systém framework Prisma. Ten pro svou funkci potřebuje znát datový model databáze. Tento datový model můžeme retrospektivně vygenerovat z již existující databáze nebo vytvořit zcela nový. V takovém případě se Prisma postará o vytvoření struktury databáze. Jelikož vytváříme zcela nový systém, je nejvýhodnější přenechat tvorbu struktury databáze na frameworku Prisma.

Pro tvorbu datového modelu frameworku Prisma se využívá podmnožiny SDL pro GraphQL. Základními stavebními bloky datového modelu jsou typy, vztahy a direktivy. Typy reprezentují entity našeho systému, každý typ je mapován na tabulku v databázi. Vztahy popisují propojení mezi těmito typy. Direktivy slouží jako pomocný nástroj pro vytvoření často využívané funkcionality, jako např. @id pro primární klíč, @unique pro unikátní klíč a další [9].

Na základě datového modelu Prisma vygeneruje tzv. klienta, což je ve své podstatě objekt, který obsahuje veškeré CRUD operace nad daným datovým modelem. Jeden klient vždy slouží pouze pro jednu databázi. Pokud však v systému potřebujeme komunikovat s větším počtem databází, můžeme mít více datových modelů, a následně klienta zvlášť pro každý model. Prisma rovněž umožňuje zvolit si jazyk, který chceme pro generování klienta využít. Mezi podporované jazyky patří JavaScript, TypeScript a Go [10].

6.1.2 GraphQL

Základem této části je server GraphQL Yoga [7], což je GraphQL server postavený na frameworku Express [8]. Důvodem pro využití serveru GraphQL Yoga je to, že s minimálním nastavením zajistí všechny funkce potřebné pro vytvoření a konfiguraci GraphQL serveru, které bychom jinak museli provést manuálně.

Zde by šlo jistě využít schéma vytvořené samotným frameworkem Prisma. Obvykle však toto schéma nechceme přímo zveřejňovat, jelikož obsahuje veškeré CRUD operace nad naší databází, což by mohlo představovat potencionální nebezpečí pro uložená data. Další nevýhodou je také to, že by nedošlo k oddělení jednotlivých vrstev. Schéma konzumované klienty by tak bylo přímo závislé na schématu databáze. Pokud bychom do schématu databáze následně chtěli přidat další sloupec, muselo by dojít k úpravě klientů. Stejně tak přidáním další hodnoty ke klientským dotazům by muselo dojít k úpravě databáze.

Další nedílnou součástí každého GraphQL serveru jsou dotazy. Ty mohou mít jeden ze tří základních typů, a to query, mutation a subscription. Queries jsou dotazy určené pouze k získání dat, mutations naopak dotazy určené ke změně dat a subscriptions slouží pro nastavení aktivního upozornění o změnách, jež se na odehrávají na serveru. Každý z těchto dotazů má přiřazený tzv. resolver. Resolver je metoda, jejíž úkolem je vykonat požadovanou operaci a navrátit její výsledek. V resolverech se následně odehrává mapování potřebné pro přechod mezi vystaveným GraphQL schématem a schématem vygenerovaným frameworkem Prisma [11].

6.1.3 Multimediální soubory

Serverová část musí mimo jiné zajišťovat i uchování multimediálních souborů. Při nahrání nového multimediálního souboru dojde k vygenerování náhodného řetězce o délce 40 znaků, pod jehož názvem se soubor uloží. Ačkoliv je vygenerování dvou stejných řetězců velice nepravděpodobné, samozřejmostí je i ověření, zda soubor s tímto jménem již neexistuje, aby nedošlo k nechtěnému přepsání již existujícího souboru. Po uložení souboru na disk se v databázi vytvoří patřičný záznam, ve kterém je uložena pouze cesta k tomuto souboru. Aby bylo možné multimediální soubory ze serverové části i získat, je využit statický middleware pro express, který zajišťuje servírování statických souborů.

6.2 Webová část

Pro vývoj administrátorského rozhraní aplikace jsem se rozhodl použít technologii React. Její hlavní výhody jsem již popsal v kapitole 5.1. Ze stejných důvodů, jako je tomu u serverové části aplikace, jsem se rozhodl doplnit React navíc o technologii TypeScript.

[7] Server GraphQL Yoga je dostupný z https://github.com/prisma-labs/graphql-yoga
[8] Framework Express je dostupný z https://expressjs.com

6.2.1 Komunikace se serverovou částí

Klíčovým prvkem celého administrátorského rozhraní je potřeba komunikace se serverovou částí. Pro tyto účely vystavuje serverová část GraphQL koncový bod. Při použití GraphQL je velice výhodné využít některého z již implementovaných GraphQL klientů. Takový klient za nás dokáže obstarat samotnou komunikaci s koncovým bodem i udržovaní veškerých stavů důležitých pro poskytování zpětné vazby uživateli. Také se postará o obnovu uživatelského rozhraní.

GraphQL klientů schopných nám tuto funkcionalitu poskytnout je celá řada. Při vývoji této aplikace nás však zajímají pouze ty, jež jsou navrženy pro práci s Reactem. Jedním z těchto klientů je Apollo Client [9]. Ten mimo jiné podporuje TypeScript a řadu dalších užitečných funkcí, jako např. využití React Hooks. Díky tomu umožňuje naplno využít výhod moderního vývoje v Reactu a je ideální volbou pro tento projekt [12].

Pro využití veškerých výhod, jež nám poskytuje TypeScript, potřebuje znát datové typy využívané vystaveným GraphQL API. Pro získání těchto datových typů můžeme použít některý z GraphQL generátorů. Takových generátorů existuje velké množství, avšak funkce většiny z nich je ve své podstatě totožná. Na základě GraphQL schématu, které je veřejně vystaveno serverovou částí sytému, jsou tyto generátory schopny vygenerovat použité datové typy pro konkrétní požadovaný programovací jazyk.

6.2.2 Uživatelské rozhraní

Častým nedostatkem webových aplikací je jejich nekonzistentnost v uživatelském rozhraní. Jednou z možností, jak tomuto problému předejít, je použití frameworku určeného pro tvorbu uživatelského rozhraní. Jedním z nejpopulárnějších UI frameworků pro React je Material-UI [10]. Ten poskytuje širou škálu již naimplementovaných a odladěných Reactových komponent, jejichž využití nám umožní rychlejší zhotovení aplikace.

Material-UI samozřejmě poskytuje způsob, jakým přizpůsobit vzhled využívaných komponent požadavkům našeho systému. Pro tyto účely se využívá tzv. thememing. Ten funguje na jednoduchém principu. Material-UI využívá pro stylování sadu přednastavených konstant z výchozího Theme objektu. Tyto konstanty lze samozřejmě předefinovat jinými hodnotami, čímž také dojde ke změně vzhledu aplikace.

To samotné však v některých případech nemusí být zcela dostačující. Pokud v aplikaci vytváříme i vlastní komponenty nebo potřebujeme provést velmi specifické úpravy využívaných komponent, potřebujeme vhodný způsob tvorby CSS stylů v JavaScript kódu.

React nám samozřejmě umožňuje napsat inline CSS styl, který lze následně aplikovat na požadovanou komponentu, avšak to je pro rozsáhlejší stylování velmi nepraktické. Dalším problémem inline CSS stylů je to, že nepodporují ani tak stěžejní funkce, jako je např. zanořovaní

[9] Knihovna Apollo Client je dostupná z https://www.apollographql.com
[10] Framework Material-UI je dostupný z https://material-ui.com

CSS stylů, práce s Media Queries nebo třeba aplikování stylů na pseudotřídy. To vše umožňuje využití inline CSS stylů pouze pro velmi drobné úpravy.

Z těchto důvodů jsem zvolil JavaScript knihovnu Emotion [11]. Ta, na rozdíl od inline stylování, všechny zmíněné funkce poskytuje. Příklad použití knihovny Emotion pro stylování React komponent je možné vidět ve zdrojovém kódu 1.

```
1  import styled from "@emotion/styled";
2  import Grid from "@material-ui/core/Grid";
3
4  export const ContentWrapper = styled(Grid)`
5      width: 100%;
6      padding: ${props => props.theme.spacing(1)};
7
8      &:after {
9          content: "";
10         display: block;
11     }
12 `;
```

Výpis 1: Příklad stylování React komponent s použitím knihovny Emotion

6.2.3 Formuláře

Hlavním účelem administrátorského rozhraní je zajištění snadné manipulace s daty. Pro tento účel jsou naprosto nepostradatelné formuláře. React samozřejmě zajišťuje základní funkcionalitu formulářů, ale jejich použití bez podpůrných knihoven není příliš praktické.

Největší překážkou při tvorbě formulářů je potřeba manuálního udržování stavů. Proto jsem se rozhodl použít knihovnu Formik [12], která za nás zajišťuje udržování stavů, jako jsou např. values, errors, touched, dirty, isValid, isValidating, isSubmitting, submitCount a mnoho dalších. Manuální udržování těchto stavů pro všechny formuláře a jejich položky by bylo nejen extrémně nepřehledné, ale zároveň velice zdlouhavé.

Formik však neumí tuto funkcionalitu zajistit zcela bez našeho přičinění. Aby byl Formik schopen udržovat stav jednotlivých položek ve formuláři, je potřeba tyto položky napřed zabalit do komponenty Field. Proto jsem pro všechny základní položky používané ve formulářích vytvořil samostatné komponenty. V nich jsem zabalil input komponentu z Material-UI do komponenty Field a následně namapoval jejich rozhraní. Výsledkem jsou znovupoužitelné komponenty TextInput, SelectInput, NumberInput, DateInput a FileInput, které zajišťující veškerou funkcionalitu potřebnou pro použití ve Formik formulářích.

[11] Knihovna Emotion je dostupná z https://emotion.sh/docs/introduction
[12] Knihovna Formik je dostupná z https://jaredpalmer.com/formik

6.2.4 Zpětná vazba

Každá správná aplikace by měla klást důraz na poskytnutí relevantní zpětné vazby. Poskytnutí zpětné vazby, například při provedení určité akce, pomůže uživatelům lépe rozpoznat, zda byla akce úspěšná či nikoliv. Jednou z možností, jak zpětnou vazbu poskytnout, je s pomocí notifikací.

Přesně pro tyto účely poskytuje Material-UI komponentu Snackbar, jejíž účelem je objevit se na obrazovce v momentě, kdy je uživateli třeba sdělit krátkou zprávu, jako např. úspěch akce, náhlou chybu, varování nebo jinou informaci. Způsob, jakým je třeba tyto komponenty vytvářet, však není zcela praktický. Z tohoto důvodu jsem využil knihovnu Notistack [13], která staví na Material-UI a zajišťuje řazení vytvářených notifikací do fronty a jejich následné zobrazení.

Pro snadnější zobrazování a schovávání notifikací jsem vytvořil komponentu NotificationProvider. Tato komponenta využívá již zmíněnou knihovnu Notistack a zajišťuje její počáteční konfiguraci. Dále rozšiřuje vytvářený Snackbar o tlačítko s ikonou křížku, které slouží ke schování notifikace. Na závěr ještě poskytuje React Hook useNotification, který nabízí metody na snadné vyvolání a schování notifikace. Příklad vytvoření notifikace s použitím useNotification lze vidět v ukázce zdrojového kódu 2.

```
1 const { notify, unnotify } = useNotification();
2 notify({message: "Při přihlašování nastala chyba", variant: "error"});
```

Výpis 2: Příklad vytvoření nové notifikace

6.3 Mobilní část

Mobilní aplikace je částí systému, které je při implementaci nutné věnovat největší pozornost. To je potřeba především proto, že mobilní aplikace bude uživatelům ve výsledku nejvíce na očích. Většina uživatelů si tak zřejmě ani neuvědomí, co vše se skrývá na jejím pozadí. Celkový dojem uživatelů se tedy bude odvíjet zejména od vzhledu a funkcí mobilní aplikace. Cílovou platformou mobilní aplikace je platforma Android. Ta nativně podporuje vývoj ve dvou programovacích jazycích. Prvním z nich je známější jazyk Java a druhým je jazyk Kotlin. Pro implementaci mobilní aplikace jsem se rozhodl využít programovací jazyk Java.

6.3.1 Komunikace se serverovou částí

Stejně, jako u webové části, je i zde potřeba zajistit komunikaci s vystaveným GraphQL koncovým bodem. K tomu je opět vhodné využít klienta, jež na základě schématu vygeneruje potřebné modely. Pro tento účel jsem použil knihovnu Apollo GraphQL Client [14] pro Android. Apollo při sestavení projektu vygeneruje typově bezpečný přístup ke koncovému bodu. Součástí knihovny je i třída ApolloClient, která slouží k samotnému vyvolání jednotlivých dotazů.

[13] Knihovna Notistack je dostupná z https://iamhosseindhv.com/notistack
[14] Knihovna Apollo GraphQL Client je dostupná z https://github.com/apollographql/apollo-android

Jednou z nejproblémovějších částí mobilní aplikace je zajištění přítomnosti dat potřebných pro funkcionalitu aplikace. Tento problém vyplývá především z nedostatečného pokrytí hradu mobilní sítí, což jsem popsal v kapitole 3.1. Kvůli tomu nám nezbývá jiné řešení než stáhnout předem všechna data a uložit je v mobilním zařízení. Pro uložení těchto dat jsem využil SQLite databázi ve spojení s interním uložištěm pro ukládání multimediálních souborů.

Při každém spuštění aplikace dojde k vyvolání dotazu CheckForUpdates na vystavený koncový bod. Dotazu se jako parametr předává datum a čas poslední aktualizace a zvolený jazyk aplikace. Dotaz následně vrátí pouze ty záznamy, které mají shodný jazyk, a jejichž datum poslední změny je větší než datum aktualizace předané v parametru. Tím je zajištěna aktuálnost dat v aplikaci, a rovněž to, že při každém spuštění nebude docházet k opětovnému stažení veškerých dat aplikace. Vždy se tedy aktualizuje pouze to, co se opravdu změnilo.

Takto však dochází pouze k aktualizaci dat z databáze. Již jsem zmínil, že v databázi nejsou uloženy multimediální soubory, ale pouze cesty k nim. Tyto soubory je tedy třeba získat jiným způsobem. Pokud jsou součástí dat vracených z dotazu CheckForUpdates i multimediální soubory, jejich cesty se předají knihovně Ion [15], která zajistí jejich stažení. Současně se vypočítá celková velikost všech stahovaných souborů a dojde ke zobrazení dialogu, ve kterém je znázorněn aktuální stav stahování.

6.3.2 Volba jazyka

Při úplně prvním spuštění aplikace je uživatel mimo jiné vyzván i k výběru jedno z dostupných jazyků. Poté, co tak učiní, dojde k uložení zvoleného jazyka do Shared Preferences, a následně ke stažení veškerých potřebných dat. Jak již bylo zmíněno v kapitole 6.3.1, aplikace ověřuje přítomnost aktualizací pomocí dotazu checkForUpdates, ve kterém mimo jiné předává zvolený jazyk. Proces stažení dat probíhá vždy stejně, bez ohledu na to, jaký jazyk si uživatel zvolí. Zvolený jazyk může uživatel změnit v nastavení aplikace. Při změně jazyka dojde ke smazaní veškerých uložených dat a následně k opětovnému vyvolání dotazu checkForUpdates. V aplikaci jsou tedy vždy uloženy pouze ta data, která jsou potřebná pro současný jazyk.

Zde je nutno zmínit, že napříč veškeré snaze minimalizovat nutnost zásahu do zdrojového kódu aplikace, se tomu v případě přidávání podpory pro nové jazyky nevyhneme. V aplikaci se totiž nachází i statické řetězce, které bez zásahu do zdrojového kódu nelze snadno přeložit. Jsou zde však učiněny potřebné kroky k tomu, aby celý tento proces mohl proběhnout co nejsnadněji. Všechny statické řetězce zobrazované v aplikaci jsou uloženy v adresáři res/values/strings. Díku tomu stačí pro přeložení těchto řetězců vytvořit nový XML soubor a následně přidat překlady pro všechny potřebné klíče.

[15] Knihovna Ion je dostupná z https://github.com/koush/ion

6.4 Oprávnění

Složitější aplikace často potřebují pro svůj provoz určité oprávnění. Veškerá oprávnění využívaná v aplikaci musí být uvedeny v aplikačním manifestu. To zahrnuje i oprávnění potřebné pro knihovny, které aplikace využívá. Tato oprávnění jsou klasifikována do několika skupin, podle toho jakou představují hrozbu pro uživatele. Běžná oprávnění, jež představují malou hrozbu, jsou udělena automaticky a nevyžadují žádnou další akci uživatele. Mezi tyto oprávnění patří například přístup k internetu nebo vibraci, které využívá i tato mobilní aplikace [13].

Oprávnění, která mohou představovat hrozbu pro uživatele, je potřeba získat za běhu aplikace a jejich udělení musí schválit sám uživatel. V dřívějších verzích Androidu stačilo tyto oprávnění schválit při instalaci aplikace. To se však ukázalo jako nedostatečné řešení, jelikož jedinou volbou uživatele bylo přijmout vše, nebo odmítnout instalaci. Toho rovněž zneužívali vývojáři některých aplikací, kteří často požadovali oprávnění nesouvisející se zaměřením jejich aplikace.

Od Androidu verze 6.0 Marshmallow je nutné o oprávnění požádat za běhu aplikace a uživatel má vždy právo toto oprávnění odmítnout. Uživatel si tedy může přesně navolit, která oprávnění aplikaci udělí, a ta se tomu musí přizpůsobit. Tato aplikace využívá dvě oprávnění vyžadující schválení uživatelem. Jedno z nich je přístup ke kameře zařízení potřebné pro skener QR kódů a druhým oprávněním je přístup k poloze zařízení. Aplikace bude nadále funkční, i pokud uživatel některé z požadovaných oprávnění odmítne, avšak funkcionalita, která je na těchto oprávněních závislá, nebude v aplikaci přístupná [13].

Žádost o oprávnění za chodu aplikace je poměrně komplikovanou záležitostí. Může totiž nastat hned několik možností, na které je třeba reagovat a ošetřit je. Pro usnadnění tohoto procesu jsem se rozhodl využít knihovnu Dexter [16], jež zajistí funkcionalitu spojenou se samotnou žádostí o oprávnění a následné vykonání metod zpětného volání onPermissionGranted, onPermissionDenied, onPermissionRationaleShouldBeShown.

6.4.1 Čtečka QR kódů

Jedna z metod pro orientaci na hradě, kterou má mobilní aplikace umožňovat, je založena na principu načítaní QR kódů. Čtečku QR kódů je možné zobrazit kliknutím na ikonu v pravé horní části obrazovky nebo výběrem patřičné položky z hlavní nabídky aplikace. Základem celé čtečky je fragment, který ve svém rozložení využívá CodeScannerView z Android knihovny Code Scanner [17]. CodeScannerView za nás obstará zobrazení náhledu kamery a detekci případných QR kódů vyskytujících se v tomto náhledu. Této funkcionality můžeme poměrně snadno dosáhnout sami s využitím SurfaceView a knihovny BarcodeDetector poskytované vývojáři Androidu. Takto fungovala čtečka QR kódů v alfa verzi mobilní aplikace.

Zde však můžeme narazit na problémy s velikostí zobrazovaného náhledu, ten může být ve výchozím stavu zploštěný a je třeba ho upravit pro konkrétní velikost zobrazení. Stejně tak se

[16] Knihovna Dexter je dostupná z https://github.com/Karumi/Dexter
[17] Knihovna Code Scanner je dostupná z https://github.com/yuriy-budiyev/code-scanner

situace začíná komplikovat v momentě, kdy chceme náhled kamery doplnit o hledáček. Proto jsem se rozhodl zvolit knihovnu Code Scanner, která je již hotovým dobře otestovaným řešením a umožňuje snadnou konfiguraci požadovaných vlastností a vzhledu.

Jakmile CodeScannerView rozpozná přítomnost QR kódu, dojde k vykonání metody zpětného volání onDecoded. Zde je třeba z detekovaného QR kódu vytáhnout textová data a následně provést jejich parsování. Každý QR kód hradu Sovinec může odkazovat na zájmový bod, trasu nebo záznam. Součástí QR kódu musí být i ID prvku, jehož detail se má zobrazit. Pokud textový obsah nelze parsovat, nebo se prvek s daným ID nenachází v SQLite databázi, dojde ke zobrazení hlášky s nápovědou v dolní části obrazovky.

6.4.2 Geolokace

Jednou z fyzických dispozicí hradu Sovinec je velmi dobrá přesnost při určování polohy uživatele. To se snaží uplatnit i tato mobilní aplikace. K získání polohy mobilního zařízení je využita knihovna EasyWayLocation [18]. Ta za nás pohlídá většinu okolních situací, které by jinak bylo potřeba manuálně ošetřit. Při použití knihovny můžeme nastavit interval pro opětovné zjištění polohy. Pro potřeby této aplikace jsem interval nastavil na 1 vteřinu.

Jakmile nám knihovna poskytne aktuální polohu mobilního zařízení, je potřeba zjistit vzdálenost k nejbližšímu zájmovému bodu. Vzdálenost k jednotlivým zájmovým bodům vypočítáme snadno využitím metody distanceTo třídy Location. Stačí tedy pouze projít všechny body a zapamatovat si ten s nejkratší vzdáleností.

Pokud je vzdálenost k nejbližšímu bodu kratší než 15 metrů, zobrazí aplikace uživateli v horní části obrazovky informaci o tom, že se nachází v blízkosti zájmového bodu, a zda si přeje zobrazit informace o daném místě. Funkcionalita je takto vytvořena proto, aby aplikace uživatele sama nepřepínala v momentě, kdy se věnuje jiné činnosti a nedocházelo tak ke zmatení uživatele. Vzdálenost 15 metrů by měla být dostatečná pro spolehlivé určení, zda se uživatel nachází v blízkosti bodu, tuto hodnotu je však možno přizpůsobit v nastavení aplikace.

6.4.3 Audioprůvodce

Jedním z požadavků na mobilní aplikaci je, aby poskytovala uživatelům možnost přehrání audioprůvodce. To mohou uživatelé udělat po zobrazení detailu záznamů. Počet zvukových nahrávek příslušných k jednotlivým záznamům není nijak omezen, což zajišťuje možnost flexibilnější úpravy obsahu. Některé body se totiž mohou zobrazovat jen dočasně, např. v průběhu expozice. Pro tyto body samozřejmě není finančně výhodné vytvářet nové zvukové nahrávky. Naopak jiné body mohou mít nahrávek více, např. stručnou a detailnější verzi.

Při přehrávání zvukové nahrávky se v dolní části obrazovky zobrazí kontrolní panel sloužící pro ovládání audio přehrávače. Tento panel je zobrazen po dobu přehrávání zvukové nahrávky, a to i v případě přechodu mezi jinými fragmenty. Přehrávání nahrávky bude pokračovat i v

[18] Knihovna EasyWayLocation je dostupná z https://github.com/prabhat1707/EasyWayLocation

případě přepnutí do jiné aplikace nebo uzamčení mobilního telefonu. Díky tomu může uživatel volně prohlížet obsah aplikace nebo se věnovat jiným aktivitám a zároveň stále poslouchat audioprůvodce.

Po dobu přehrávání nahrávek je navíc přidána notifikace umožňující ovládání audio přehrávače. Tato notifikace se zobrazuje v notifikačním panelu i na zamčené obrazovce mobilního zařízení, čímž umožňuje uživateli snadné ovládání přehrávače bez nutnosti otevření aplikace.

6.4.4 Obrázková galerie

Obrázková galerie by měla být nedílnou součástí každé aplikace, která ve větší míře pracuje z obrázky. Proto si našla své uplatnění i v této mobilní aplikaci. Implementace obrázkové galerie je provedena v separátní aktivitě. To je nutné proto, abychom mohli dosáhnout efektu překrytí obrazovky. Takového efektu si můžeme všimnout např. u dialogů, kdy obsah překrytý dialogem je stále viditelný, avšak dojde k jeho zatmavění.

(a) Jednotlivé vrstvy obrázkové galerie

(b) Výsledný efekt

Obrázek 6: Obrázková galerie

Struktura aktivity ImageGalleryActivity je složena z několika různých vrstev, které společně tvoří výsledný efekt galerie. Základem této aktivity je zobrazení ViewPager2 z podpůrné knihovny AndroidX. Mimo to aktivita obsahuje ještě TextView zobrazující popisek obrázku a zobrazení WormDotsIndicator z knihovny Material View Pager Dots Indicator [19], které indikuje

[19]Knihovna Material View Pager Dots Indicator je dostupná z https://github.com/tommybuonomo/dotsindicator

celkový počet obrázků v galerii. Jednotlivé vrstvy aktivity a výsledný efekt galerie je možno vidět na obrázcích 6a a 6b.

Zobrazení ViewPager2 obsahuje fragmenty ImageGalleryPageFragment, mezi kterými se lze přesouvat pomocí gesta swipe. Každý tento fragment obsahuje zobrazení SubsamplingScaleImageView ze stejnojmenné knihovny Subsampling Scale Image View [20]. Toto zobrazení za nás obstarává veškerou funkcionalitu spojenou s přibližováním obrázku a zároveň poskytuje mechanizmy pro zobrazování velkých bitmap.

Při zobrazení galerie je potřeba počítat s tím, že může obsahovat několik bitmap ve velkém rozlišení, a tím také klást velké nároky na paměť mobilního zařízení. Z toho důvodu jsou ve fragmentu ImageGalleryPageFragment využity metody životního cyklu onResume a onPause. K načtení obrázku dochází teprve v metodě onResume a všechny prostředky jsou opět uvolněny v metodě onPause. Tím je limitováno množství paměti potřebné pro procházení obrázkové galerie.

6.4.5 Kvíz

Pro zpestření zážitku z aplikace je uživatelům poskytnuta možnost zahrát si kvíz. Před spuštěním kvízu si uživatel může zvolit jednu ze tří obtížností. Podle zvolené obtížnosti následně dojde k vygenerování patřičných kvízových otázek. Každá otázka může mít jednu nebo více správných odpovědí, což umožňuje vytvářet důmyslnější kvízové otázky.

Pro zobrazení kvízu je využit fragment QuizDetailFragment. V případě nové hry fragment zajistí vygenerování náhodných kvízových otázek. Navíc je možno mu prostřednictvím argumentu předat objekt QuizHistoryHolder. Všechny odehrané hry, včetně vygenerovaných otázek, se totiž ukládají do SQLite databáze. To proto, aby si uživatel mohl znovu zkusit zahrát kvíz, který se mu nepovedl. Při opětovném průchodu dojde k načtení otázek z objektu QuizHistoryHolder, čímž se zajistí, že otázky v kvízu budou stejné. Aby si však uživatel pouze nezapamatoval pořadí odpovědí, vždy před zahájením kvízu dojde k promíchání pořadí kvízových otázek a jejich odpovědí.

Fragment QuizDetailFragment zajišťuje i logiku pro přepínání otázek a zobrazení konečného hodnocení. V jádru fragmentu je použito zobrazení ViewPager2, jež obsahuje jednotlivé otázky. Každá otázka je zabalena do fragmentu QuizQuestionPageFragment, který zajišťuje zobrazení otázky a odpovědí, odpočítávání času a samotné vyhodnocení otázky. Při vyhodnocení každé otázky dojde ke znázornění správnosti jednotlivých odpovědí.

[20] Knihovna Subsampling Scale Image View je dostupná z https://github.com/davemorrissey/subsampling-scale-image-view

7 Testování

Každou aplikaci je před jejím nasazením vhodné otestovat na co největším vzorku různých zařízení. Testování je vhodným nástrojem nejen k odhalením chyb, které se v aplikaci mohou skrývat, ale zejména k získání tížené zpětné vazby od samotných uživatelů aplikace. Uživatelé navíc sami často přichází s nápady na nové funkce a vylepšení, které by aplikace mohla nabízet. To může zásadní mírou přispět k dalšímu rozvoji aplikace.

První rozsáhlejší testování aplikace proběhlo dne 22. února 2020 na hradě Sovinec. Do testování se zapojilo zhruba patnáct dobrovolníků z řad fanoušků hradu. V této době nebyla mobilní aplikace ještě nasazena na žádné distribuční službě, z toho důvodu musela být distribuována s využitím fyzické instalace APK balíčku na samotných mobilních zařízeních. Při testování bylo v aplikaci odhaleno několik chyb a zároveň se povedlo získat i náměty na možná vylepšení.

Nejčastější chybou, na kterou uživatelé při testování naráželi, byl pád aplikace. K němu mohlo dojít na několika různých místech a měl vesměs dvě hlavní příčiny. První z nich je špatné stažení datových souborů při aktualizaci aplikace. To bylo vyřešeno opětovným stažením dat v momentě, kdy došlo k přerušení jejich stahování. Druhou příčinou bylo využití špatného vlákna při změně uživatelského rozhraní, což bylo vyřešeno důslednějším přepínáním na hlavní vlákno v momentě, kdy mělo dojít ke změně uživatelského rozhraní.

Další chybou, která se v průběhu testování vyskytla, bylo problematické určování polohy mobilního zařízení. Určování polohy bylo mimo GPS závislé ještě na připojení k mobilní síti. To vedlo k tomu, že v místech s nedostupným pokrytím nebyly uživateli nabízeny doporučení na základě polohy. Chyba byla opravena výměnou knihovny poskytující informace o poloze za jinou, která nemá určování polohy závislé na připojení k mobilní síti.

Jedním z námětů na vylepšení, které byly do aplikace zapracovány, bylo doplnit čtečku QR kódů o výřez s rámečkem. Pokud by se QR kód nacházel mimo tento výřez, nemělo by dojít k jeho naskenování. Doposud čtečka skenovala všechny QR kódy, které se vyskytly v záběru kamery. Dalším zapracovaným námětem bylo přidání odkazu na čtečku QR kódů do hlavní nabídky aplikace. Do té doby bylo možné přejít na čtečkou pouze pomocí ikony v horním panelu aplikace. Posledním zapracovaným námětem na vylepšení bylo přidání notifikace přehrávače. Přehrávání je tak možné ovládat přímo v panelu notifikací nebo například ze zamčené obrazovky.

8 Nasazení

Jak již bylo zmíněno, součástí systému jsou tři hlavní celky, kdy pro každý z nich bude nutno zajistit jeho nasazení. Nejprve je potřeba nasadit serverovou a webovou část systému. To samotné ovšem pro celkové zajištění chodu systému stačit nebude. Serverová část využívá databázový framework Prisma, který ke své činnosti zase potřebuje přístup k databázi.

Takový proces nasazování může být poměrně zdlouhavou záležitostí a je při něm potřeba myslet na mnoho specifických detailů, které jsou doménou zejména odborníků na DevOps. Pro usnadnění celého procesu jsem se rozhodl využít Docker [21]. Ten umožňuje jednotlivé aplikace zabalit do samostatných celků a ty následně nezávisle na sobě spustit jako tzv. kontejnery. Samotné kontejnery běží přímo v kernelu daného zařízení a díky jejich výborné izolaci je možno spustit více kontejnerů na jediném hostu. Tím odpadá potřeba pro virtuální stroje [14].

Proto, abychom mohli zabalit serverovou a webovou část do samostatných celků, potřebujeme specifikovat konfigurační soubor Dockerfile. V něm se krok za krokem specifikují všechny příkazy, které je třeba provést pro zabalení aplikace. Následně již stačí spustit příkaz docker build, který nám sestaví výsledný image. Framework Prisma a databáze MySQL, kterou budeme používat, již mají vlastní image předpřipraven, takže ty stačí pouze stáhnout.

Pokud chceme spouštět všechny tyto aplikace společně, je vhodné si k tomu připravit soubor Docker Compose. V něm můžeme provést veškerou konfiguraci tak, abychom při dalším nasazení nemuseli znovu konfigurovat každý image. Zjednodušený příklad souboru Docker Compose je uveden na straně 41 v příloze A [15, 16].

Poslední částí, kterou je potřeba nasadit, je samotná mobilní aplikace. Pro distribuci aplikací na platformě Android je vhodné zvolit některou z dostupných distribučních služeb. Bezesporu nejpopulárnější distribuční službou pro platformu Android je služba Google Play. Pro účely distribuce této aplikace poskytla katedra informatiky přístup k účtu služby Google Play, na kterém bude aplikace zveřejněna.

[21] Docker je dostupný z https://www.docker.com

9 Závěr

Výsledkem této bakalářské práce je produkt, který prošel všemi fázemi vývoje, a je připraven pro ostrý provoz. Při vývoji systému jsem dbal na vytvoření implementace, která je nezávislá na jeho konkrétním použití. Systém tedy není pevně vybudován pro účely hradu Sovinec, ale je možné jej relativně snadno adaptovat na průvodce jakékoliv jiné lokality.

Systém byl navržen s důrazem na snadnou manipulaci obsahu. Veškerá data, která se v mobilní aplikaci zobrazují, nejsou napevno zabudována v jejím zdrojovém kódu. Všechny nástroje potřebné pro úpravu obsahu aplikace jsou poskytnuty formou přehledného webového administrátorského rozhraní. Díky tomu mohou administrátoři i bez hlubších znalostí o tom, jak systém na pozadí funguje, udržovat obsah aplikace stále aktuální.

Vytvořená mobilní aplikace je postavena na silných stránkách konkurenčních produktů, klade maximální důraz na uživatelskou přívětivost a disponuje rozsáhlou škálou funkcí, které se v podobných aplikacích hledají pohromadě obtížně. Proto, aby bylo možné aplikaci zpřístupnit co největšímu počtu případných uživatelů, je výsledná aplikace optimalizována pro Android verze 5.0 a vyšší. Tyto verze dohromady pokrývají přes 94 procent všech mobilních Android zařízení.

I přes to, že jsem se snažil do aplikace zapracovat co největší množství funkcí a udělat ji tak co nejatraktivnější, stále v aplikaci spatřuji prostor na vylepšení. Jednou z funkcí, kterou bych rád v budoucnu do aplikace přidal, je hlubší propojení se sociálními sítěmi. Uživatelé by tak mohli např. sdílet zajímavé momenty z aplikace se svými přáteli, což by hradu Sovinec pomohlo získat větší pozornost a tím i větší množství návštěvníků. Toho by šlo relativně snadno docílit např. využitím frameworku Facebook SDK [17].

Tvorba této práce mě také přivedla k myšlence, zda zvolení Androidu jako cílové platformy bylo tou správnou volbou. V aplikaci vidím značný komerční potenciál, ten je však tímto omezen pouze na platformu Android. To by se samozřejmě dalo vyřešit tak, že bych vytvořil stejnou aplikaci i pro platformu iOS. Tento přístup s sebou ovšem nese následné udržování dvou různých aplikací, což je při vývoji jednotlivcem, popřípadě malým týmem, příliš velká zátěž.

Z tohoto důvodu bych v budoucnu upřednostnil vývoj na frameworku, který, až na specifické výjimky, zajišťuje jednotnou implementaci pro obě platformy. Zejména u tohoto projektu by se nejvíce hodilo použití frameworku React Native [22]. Ten pracuje s JavaScriptovou UI knihovnou od vývojářů Facebooku a napsaný kód je následně sestaven na nativní Android a iOS aplikaci. To by mimo jiné přineslo sjednocení vývoje napříč serverovou, webovou a mobilní částí systému, takže veškerý vývoj by mohl zajistit jediný vývojář se znalostí JavaScriptu [18].

[22] Framework React Narive je dostupný z https://reactnative.dev

Literatura

1. *Navigation drawer* [online]. Google [cit. 2020-05-02]. Dostupné z: https://material.io/components/navigation-drawer.
2. *Introduction* [online]. Prisma [cit. 2020-04-02]. Dostupné z: https://www.prisma.io/docs/understand-prisma/introduction.
3. *Introduction to React* [online]. SurviveJS [cit. 2020-03-31]. Dostupné z: https://survivejs.com/react/getting-started/introduction-to-react.
4. *Tutorial: Intro to React* [online]. Facebook [cit. 2020-03-30]. Dostupné z: https://reactjs.org/tutorial/tutorial.html.
5. *Introduction to Node.js* [online]. OpenJS Foundation [cit. 2020-04-03]. Dostupné z: https://nodejs.dev.
6. MÁCA, Jindřich. *Lekce 1 - Úvod do Node.js* [online]. ITnetwork.cz [cit. 2020-03-29]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs.
7. *Introduction to GraphQL* [online]. The GraphQL Foundation [cit. 2020-04-01]. Dostupné z: https://graphql.org/learn.
8. KHACHATRYAN, Grigor. *What is GraphQL?* [online]. A Medium Corporation [cit. 2020-04-06]. Dostupné z: https://medium.com/devgorilla/what-is-graphql-f0902a959e4.
9. *Datamodel (MySQL)* [online]. Prisma [cit. 2020-04-26]. Dostupné z: https://v1.prisma.io/docs/1.34/datamodel-and-migrations/datamodel-MYSQL-knul.
10. *Generating the Client (TypeScript)* [online]. Prisma [cit. 2020-04-26]. Dostupné z: https://v1.prisma.io/docs/1.34/prisma-client/setup/generating-the-client-TYPESCRIPT-r3c2.
11. *Execution* [online]. The GraphQL Foundation [cit. 2020-04-26]. Dostupné z: https://graphql.org/learn/execution.
12. *Introduction* [online]. Meteor Development Group [cit. 2020-04-03]. Dostupné z: https://www.apollographql.com/docs/react.
13. *Handle Runtime Permissions in Android* [online]. A Medium Corporation [cit. 2020-04-22]. Dostupné z: https://medium.com/programming-lite/runtime-permissions-in-android-7496a5f3de55.
14. *Docker overview* [online]. Docker [cit. 2020-03-29]. Dostupné z: https://docs.docker.com/engine/docker-overview.
15. *Overview of Docker Compose* [online]. Docker [cit. 2020-03-29]. Dostupné z: https://docs.docker.com/compose.
16. *Docker* [online]. Prisma [cit. 2020-03-29]. Dostupné z: https://v1.prisma.io/docs/1.34/prisma-server/deployment-environments/docker-rty1.

17. LACKO, Ľuboslav. *Mistrovství - Android*. Brno: Computer Press, 2017. ISBN 978-80-251-4875-4.

18. *React Native* [online]. Facebook [cit. 2020-04-25]. Dostupné z: https://reactnative.dev.

A Příklad souboru Docker Compose

```
version: '3'
services:
    mysql:
        image: 'mysql:5.7'
        restart: always
        environment:
            MYSQL_ROOT_PASSWORD: prisma
        ports:
            - '3306:3306'
        volumes:
            - 'mysqlvoll:/var/lib/mysql'
    prisma:
        image: 'prismagraphql/prisma:1.34'
        restart: always
        ports:
            - '4466:4466'
        environment:
            PRISMA_CONFIG: |
                port: 4466
                databases:
                    default:
                        connector: mysql
                        migrations: true
                        host: database
                        port: 3306
                        user: root
                        password: prisma
        depends_on:
            - mysql
volumes:
    mysqlvoll: null
```

Výpis 3: Zjednodušený příklad souboru docker-compose.yaml

B Licence

Toto dílo podléhá licenci Creative Commons Uveďte původ-Neužívejte komerčně 4.0 Mezinárodní License. Dílo smíte rozmnožovat, distribuovat, změnit a vyjít z původního díla. Musíte uvést autora a nesmíte dílo využívat komerčně.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. You are free to copy, redistribute, transform, and build upon the material. You must mention the author and cannot use the work for commercial purposes.